Wastage-Aware Routing in Energy-Harvesting Software Defined Wireless Sensor Networks

A DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF REQUIREMENT FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

GAZIPUR, BANGLADESH

SUBMITTED BY
Md. Rayhan Kabir (134421)
Rafid Abyaad (134407)

Supervised by
Prof. Dr. Muhammad Mahbub Alam
Head of the Department,
Department of Computer Science and Engineering,
Islamic University of Technology.

November, 2017

Declaration of Authorship

We, Rayhan Kabir (134421) and Rafid Abyaad(134407), declare that this thesis titled "**Wastage-Aware Routing in Energy-Harvesting Software Defined Wireless Sensor Networks**" and the works presented in it are our own. We confirm that:

- This work has been done for the partial fulfillment of the Bachelor of Science in Computer Science and Engineering degree at this university.
- Any part of this thesis has not been submitted anywhere else for obtaining any degree.
- Where we have consulted the published work of others, we have always clearly attributed the sources.

Submitted by:

Md. Rayhan Kabir (134421)

Rafid Abyaad (134407)

Wastage-Aware Routing in Energy-Harvesting Software Defined Wireless Sensor Networks

Approved by:

____

Prof. Dr. Muhammad Mahbub Alam

Head of the Department,

Department of Computer Science and Engineering,

Islamic university of Technology

Date:…………………………….

Abstract

Techno-economic drivers are creating the conditions for a radical change of paradigm in the design and operation of future telecommunications infrastructures. In fact, SDN, NFV, Cloud and Edge-Fog Computing are converging together into a single systemic transformation termed "Softwarization" that will find concrete exploitations in network management. Although wireless equipment manufacturers are increasing their involvement in SDN-related activities, to date there is not a clear and comprehensive understanding of what are the opportunities offered by SDN in most common networking scenarios involving wireless infrastructureless communications and how SDN concepts should be adapted to suit the characteristics of wireless and mobile communications. Here we studied different proposed protocol architecture for software defined wireless network, effective ways for energy efficient WSN and found some of the shortcomings of them. We discuss about some of the challenges facing IOT paradigm and major design requirements as well; with the intention to merge SDN and energy reservation approaches for better performance in terms of network lifetime and latency.

Acknowledgement

First and formost, we offer gratitude to the Almighty Allah (SWT) for giving us the capability to do this work with good health.

We are greatful to our thesis supervisor, Dr, Muhammad Mahbub Alam, for the support and guidance throughtout our research at Islamic University of Technology(IUT). He created a nice research environment for which we have able to explore many ideas without constraint. We have gaines a wealth of knowledge and future endeavor. For all of his eggorts as oir true mentor, we express our heartfelt gratitude to him.

We would like to thank all the faculty members of the department of CSE, IUT for their inspiration and help.

And last but not least we are thankful to our family, friends and well wishers for their support and inspiration. Withiout them it would never been possible for us to make it this far.

Table of Contents

1. INTRODUCTION

Designing and managing network has become one of the biggest challenge for the ever increasing networks. In past few years with the aid of SDN (software-defined networking) Designing and managing networks has become more innovative. This technology seems to have appeared suddenly, but it is actually part of a long history of trying to make computer networks more programmable. Software defined networking is proposed to reduce the complexity of network configuration.

Computer networks are complex and difficult to manage. They involve many kinds of equipment from routers and switches to middle boxes such as firewalls, network address translators, server load balancers, and intrusion- detection systems. Routers and switches run complex, distributed control software that is typically closed and proprietary. The software implements network protocols that undergo years of standardization and interoperability testing. Network administrators typically configure individual network devices using configuration interfaces that vary between vendors—and even between different products from the same vendor. Although some network-management tools offer a central vantage point for configuring the network, these systems still operate at the level of individual protocols, mechanisms, and configuration interfaces. This mode of operation has slowed innovation, increased complexity, and inflated both the capital and the operational costs of running a network. SDN is changing the way networks are designed and managed. It has two defining characteristics. First, SDN separates the control plane (which decides how to handle the traffic) from the data plane (which forwards traffic according to decisions that the control plane makes). Second, SDN consolidates the control plane, so that a single software control program controls multiple dataplane elements. The SDN control plane exercises direct control over the state in the network's dataplane elements (i.e., routers, switches, and other middleboxes) via a well-defined API.

Software Defined Networking (SDN) promises to dramatically reduce the complexity of network configuration and management as well as to make the introduction of innovation in the network operations possible. Accordingly, SDN design and experimentation is the subject of the increasing attention of the industrial and academic research the institution of large industry-driven organizations focused on SDN such as the Open Networking Foundation (ONF) (https://www.opennetworking.org/) witness such increasing interest. The SDN has interested the wireless networking community as well In fact, an increasing number of enterprises working in the field of wireless and mobile communications have joined SDN-related initiatives. For example, Verizon, Nokia Siemens Networks, Ericsson, and Netgear are current members of ONF. Regardless of such increasing interest, to the best knowledge, there is not a clear and comprehensive understanding of what are the advantages of SDN in the most common wireless infrastructureless networking scenarios and how the SDN concept should be expanded to suit the characteristics of wireless and mobile communications.

In case of wireless sensor networks, adopting SDN is little bit complex but it can put a remarkable improvement in routing over current routing protocols. In wireless sensor networks the nodes can be source node, target node or forwarding node. The high dynamic characteristics of wireless link cause poor quality and low stability for link, which poses a challenge to throughput and transmission reliability of wireless sensor network. Otherwise, restricted energy and mobility requirements of node also bring difficulties to design and optimization of routing protocol.

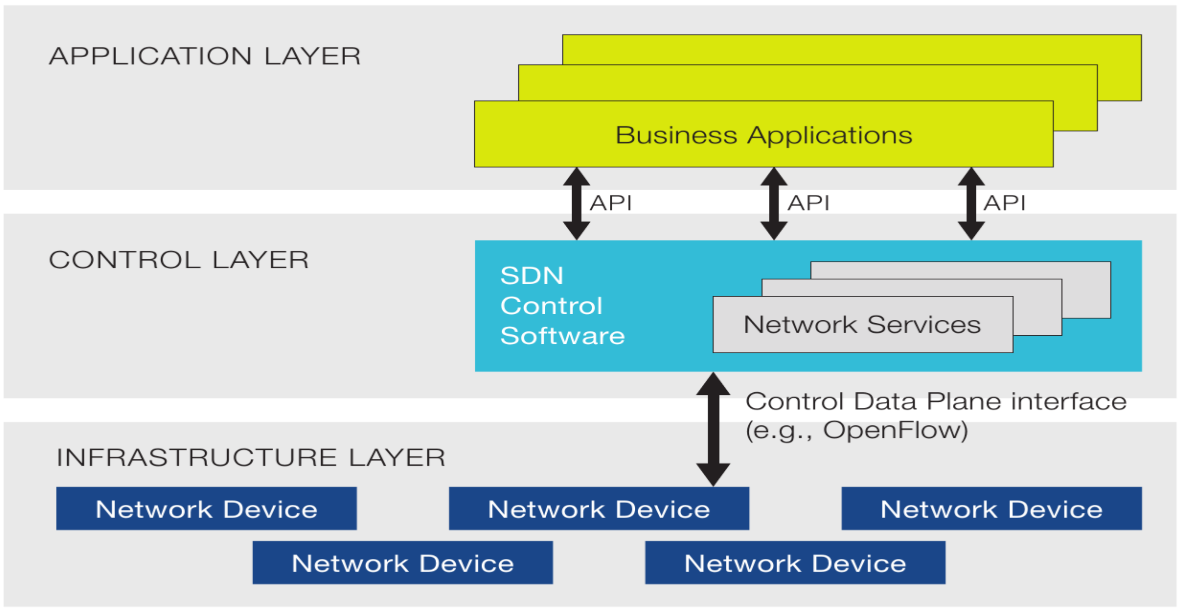


Figure: SDN architecture

Traditional multi-hop wireless routing is divided into active routing and passive routing; active routing such as OLSR{X} is based on broadcast information; in each node, the routing information from that node to all other nodes is saved, so there is so much routing information that requires to be saved in each node, and too much internal storage is occupied; therefore, active routing is not adapted to high dynamic network. As for passive routing such as AODV{X} the routing is searched with broadcast each time when sending data is required by node; when multiple nodes require sending routing, nodes need broadcasting for times to search routing; when there are too many links for a node, too much energy is consumed by broadcast. SDN separates control from data, and open uniform interface (such as OpenFlow) is adopted for interaction. Control layer is responsible for programming to manage & collocate network, to

deploy new protocols, and etc. Through centralized control of SDN, uniform network-wide view may be obtained, and dynamic allocation may be conducted to network resources as per changes in network flow [4]. Currently, the most routing researches for software-defined network are with respect to wired network and data center [5, 6]; though software-defined Internet of Things and software-defined wireless sensor network are put forth in a few researches, but only at stage of putting forth models and concepts. In researches on SDN based on wireless network, the characteristics of wireless network, such as broadcast characteristics, hidden terminal, node mobility and etc. shall be taken into consideration. OpenFlow Protocol is only applicable to route selection, however, applying more functions such as perceiving a variety of sensor data, sleep, aperiodic data collection and etc. in wireless network node, cannot be realized with OpenFlow Protocol and Standard.

2. SDN OVERVIEW

The basic idea of SDN is the separation of control plane and packet forwarding plane. In SDN control plane is programmable hence giving rise to innovation. Software Defined Networking (SDN) assure to strikingly minimize the current complex networking environment and its management. Innovation feature is making the networking environment more attractive to work on. User can differentiate between the service providers because of various dimensions which are not there in traditional networks. This will help drastically to maximize the network ability to deliver to the best and more efficiently. SDN is seen as area of attraction for researchers and hardware manufacturers in recent times. For example Ericsson, Net-gear, Nokia Siemens Networks and Verizon are in the list of current members of Open Networking Foundation (ONF). Along with what discussed above, various opportunities which should be trapped with help of implementing the SDN to wireless infrastructure-less networks are stated in the study. SDN is based on abstraction. Separation between control plane and forwarding plane is done. An interactive application environment is provided with control plane. Network topology information plays critical role with the help of which decision can be changed, modified and designed in completely new manner with help of SDN application interface. Control plane can

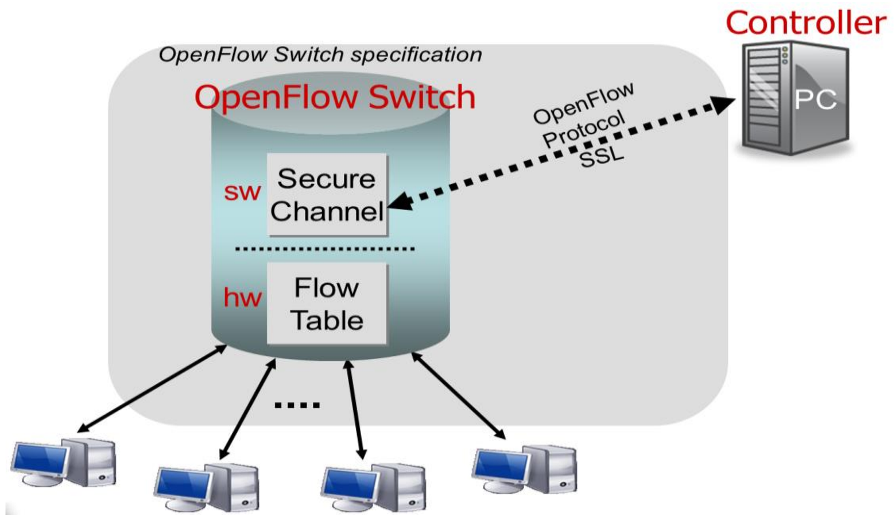


FIGURE: SDN switch specification

also be termed as the operating system of the network. Refer Figure. for more detailed SDN design concept.

A very critical task is to select a SDN model or models SDN can support. Basically there are three types of models which are network virtualization model, evolutionary model and the OpenFlow model of SDN. The most important goal of network virtualization is to get rid of restrictions on LAN partitioning that resides in Ethernet Virtual LAN standards and solve the issues like scalability & multicasting in the network architectures. One highly popular advantage of network virtualization is that it is affirmative towards multitenant clouds without making any change in the network but the most significant negative of virtualization is that it adds

complexity and extra overheads. Evolutionary model objective is to maximize software control within the boundaries of network topologies. Integration of standards in the devices by vendors is a major problem in this model. Some do it some won't there compatibility issue arises. Last one is OpenFlow is the model, on which the basic idea on SDN is built [8]. There must be a mechanism for frequent changes to a network. These change will be both in conditions and network state. There must be a high level language for network configuration support. There

must be better visibility and control on network troubleshooting tasks as well as on its diagnosis. For the solution a wide range of high level language network policies and correct scan of networproblem.

Software Defined Networking (SDN) provides very interesting features which will revolutionize the future networks like centralize control mechanism, cost efficiency, innovation, programmability, scalability, security, virtualization, cloud support, automation, reliability and

efficient environment to support Big-Data. It gives us a centralized control mechanism of various networking hardware devices from multiple vendors, further network automation is greatly improved with the help of APIs which are used to abstract network details. One of the

main objective of SDN is to provide innovation over the internet or the network, this ensure very less vendor dependence. Programmability by operators is feature of Software Defined Networking which helps to put innovation into practice. Security and reliability are the major issues in today's networks, SDN enhance these two aspects to the maximum. Simple operation and cost efficiency are key factors with openness to design and invent in SDN. SDN makes it easy to share resources in an efficient manner with the feature of network intelligence. Further paper stated SDN is helpful in service aware networking too. With SDN, Network Management has improved vastly which also give it cutting edge on traditional networking architecture and important determinant in future works.

Networks based on Software Defined Networking are being implemented both on testbed and production networks. Fault tolerance property is a key for the production networks and a most desirable & a must for SDN networks. It is noted that with respect to fault tolerance in SDN there are not much researches. A fault tolerance SDN architecture is stated which quickly recover from the occurred faults and can work on highly scalable networks. The architecture describes the recovery from multiple links failures. SDN based networks are more and more been deployed both on testbed and production networking environments.**OpenFlow:** A dedicated OpenFlow Switch is a dumb data path element that forwards packets between ports, as defined by a remote control process. Figure 1 shows an example of an OpenFlow Switch.

In this context, flows are broadly defined, and are limited only by the capabilities of the particular implementation of the Flow Table. For example, a flow could be a TCP connection, or all packets from a particular MAC address or IP address, or all packets with the same VLAN tag, or all packets from the same switch port. For experiments involving non-IPv4 packets, a flow could be defined as all packets matching a specific (but non-standard) header.

Each flow-entry has a simple action associated with it; the three basic ones (that all dedicated OpenFlow switches must support) are:

1. Forward this flow's packets to a given port (or ports). This allows packets to be routed through the network. In most switches this is expected to take place at line rate.

2. Encapsulate and forward this flow's packets to a controller. Packet is delivered to Secure Channel, where it is encapsulated and sent to a controller. Typically used for the first packet in a new flow, so a controller can decide if the flow should be added to the Flow Table. Or in some experiments, it could be used to forward all packets to a controller for processing.

3. Drop this flow's packets. Can be used for security, to curb denial of service attacks, or to reduce spurious broadcast discovery traffic from end-hosts.

An entry in the Flow-Table has three fields:

(1) A packet header that defines the flow,

(2) The action, which defines how the packets should be processed

(3) Statistics, which keep track of the number of packets and bytes for each flow, and the time since the last packet matched the flow (to help with the removal of inactive flows).

In the first generation "Type 0" switches, the flow header is a 10-tuple shown in Table 1. A TCP flow could be specified by all ten fields, whereas an IP flow might not include the transport ports in its definition. Each header field can be a wildcard to allow for aggregation of flows, such as flows in which only the VLAN ID is defined would apply to all traffic on a particular VLAN.

In port	VLAN ID	Ethernet			IP			TCP	
		SA	DA	Type	SA	DA	Port	Src	Dst

Controllers:

A controller adds and removes flow-entries from the Flow Table on behalf of experiments. For example, a static controller might be a simple application running on a PC to statically establish flows to interconnect a set of test computers for the duration of an experiment. In this case the flows resemble VLANs in current networks— providing a simple mechanism to isolate experimental traffic from the production network. Viewed this way, OpenFlow is a generalization of VLANs.

One can also imagine more sophisticated controllers that dynamically add/remove flows as an experiment progresses. In one usage model, a researcher might control the complete network of OpenFlow Switches and be free to decide how all flows are processed. A more sophisticated controller might support multiple researchers, each with different accounts and permissions,

enabling them to run multiple independent experiments on different sets of flows. Flows identified as under the control of a particular researcher (e.g., by a policy table running in a controller) could be delivered to a researcher's user-level control program which then decides if a new flow-entry should be added to the network of switches.

3. DESIGN ISSUES

3.1 Scalability:

One possible SDN design is to push all the control functionality to a centralized controller. Empowered with a complete network-wide view, developing control applications and enforcing policies become much easier in this setting. Nevertheless, controllers can potentially become the bottleneck in the network operation. As the size of a network grows, more events and requests are sent to the controller, and at some point, the controller cannot handle all the incoming requests. Early benchmarks of an SDN controller (NOX) showed that it can handle 30k requests/s. Even though this may be sufficient for a sizeable enterprise network, it could be a major problem for data-center-like environments with high flow initiation rates. One way to alleviate this concern is to level parallelism in multicore systems and improve IO performance. Tootoonchian et al. showed that simple modifications to the NOX controller boosts its performance by an order of magnitude on a single core. It means that a single controller can support a far larger network, given sufficient controller channel bandwidth with acceptable latency. We can also reduce the number of requests forwarded to the controller. DIFANE [5] proactively pushes all state to the data path In DevoFlow , with support from an ASIC, short-lived flows are handled in the data path, and only larger flows are forwarded to the controller, effectively reducing the load on the controller and improving scalability. Alternatively, one can distribute the state and/or computation of the control functionality over multiple controllers. Having a centralized controller is by no means an intrinsic characteristic of SDN. All we need is a unified networkwide view to reap the benefits of SDN. As stated in [7], it is not always feasible to achieve strong consistency while maintaining availability and partition tolerance. Therefore, selecting an apt consistency level is an important design trade-off in SDN. To preserve scalability, one should design control applications with the weakest possible consistency level. There are solutions where we can physically distribute the control plane elements, yet maintain the network-wide view. Onix [2], for example, is a distributed control platform that facilitates implementation of distributed control planes. It provides control applications with a set of general APIs to facilitate access to network state (NIB), which is distributed over Onix instances. HyperFlow. This keeps the simplicity of developing the control plane on a central controller while alleviating a class of scalability issues associated with a centralized controller, albeit for a more restricted set of control applications satisfying certain properties. Kandoo takes a different approach to distributing the control plane. It defines a scope of operations to enable applications with different requirements to coexist: locally scoped applications (i.e., applications that can operate using the local state of a switch) are deployed close to the data path in order to process frequent requests and shield other parts of the control plane from the load. A root controller, on the other hand, takes charge of applications that require network-wide state, and also acts as a mediator for any coordination required between local controllers. An interesting observation is that control plane scalability challenges in SDN (e.g., convergence and consistency requirements) are not inherently different than those faced in traditional network design. SDN, by itself, is neither likely to eliminate the control plane design complexity or make it more or less scalable.1 SDN, however:

• Allows us to rethink the constraints traditionally imposed on control protocol designs (e.g., a fixed distribution model) and decide on our own trade-offs in the design space

• Encourages us to apply common software and distributed systems development practices to simplify development, verification, and debugging Unlike traditional networks, in SDN, we do not need to address basic but challenging issues like topology discovery, state distribution, and resiliency over and over again.

As demonstrated in Onix, control applications can rely on the control platform to provide these common functions; functions such as maintaining a cohesive view of the network in a distributed and scalable fashion. In fact, it is significantly easier to develop applications for such cohesive distributed control platforms than a swarm of autonomous applications running on heterogeneous forwarding elements.

Flow Initiation Overhead — Ethane [10], an early SDN security system, puts a controller in charge of installing forwarding state on switches on a per-flow basis. Even though this reactive form of flow handling introduces a great degree of flexibility (e.g., easy fine-grained high-level network-wide policy enforcement in the case of Ethane), it introduces a flow setup delay and, depending on implementation, may limit scalability. Early designs, such as Ethane and NOX, lead to the widespread assumption that all SDN systems are reactive. In reality, however, proactive designs — in which forwarding entries are set up before the initiation of actual flows — are perfectly acceptable in SDN, and can avoid the flow setup delay penalty altogether. Let us review the flow setup process to explain the bottlenecks and show how a good design can avoid them. As illustrated in Fig. 2, the flow setup process has four steps:

• A packet arrives at the switch that does not match any existing entry in the flow table.

• The switch generates a new flow request to the controller.

• The controller responds with a new forwarding rule.

• The switch updates its flow table. The performance in the first three steps and partially the last depends on the switch capabilities and resources (management CPU, memory, etc.) and the performance of its software stack.

The delay in the third step is determined by the controller's resources along with the control program's performance. Finally, the switch's FIB update time contributes to the delay in completing the flow setup process. Assuming controllers are placed in close proximity of switches, the controller-switch communication delay is negligible. On the controller side, even on a commodity machine with a single CPU core, state-of-the-art controllers are well capable of responding to flow setup requests within a millisecond when the flow initiation requests are on the order of several hundred thousand per second. While Open vSwitch — an OpenFlow-enabled software switch — is capable of installing tens of thousands of flows per second with sub-millisecond latency, hardware switches only support a few thousand installations per second with a sub-10 ms latency at best. This poor performance is typically attributed to lack of resources on switches (weak management CPUs), poor support for high-frequency communication between the switching chipset and the management CPU, and non-optimal software implementations. We expect these issues to be resolved in a few years as more specialized hardware is built. It is foreseeable that the FIB update time will become the main factor in the switch-side flow setup latency. While we argue that controllers and, in the near future, switches would be able to sustain sufficient throughput with negligible latency for reactive flow setup, in the end the control logic determines the scalability of a reactive design. A control program installing an end-to-end path on a per-flow basis does not scale, because the perswitch memory is fixed but the number of forwarding entries in the data path grows with the number of active flows in the network. However, the control program may install aggregate rules matching a large number of micro-flows (thereby facing the same scalability challenges as a proactive design), or proactively install rules in the network core to provide end-to-end connectivity and identify quality of service (QoS) classes, while classifying and reactively labeling flows at the edge. A viable solution to the scalability challenges of the proactive designs in the former class due to data path memory scarcity is proposed in DIFANE [5]; while the scalability of the latter class follows from the observation that the fanout of an edge switch and thus the number of flows initiated there is bounded (just add edge controllers as the network grows in size). Resiliency to Failures — Resiliency to failures and convergence time after a failure have always been a key concern in network performance. SDN is no exception, and, with the early systems setting an example of designs with a single central control, resiliency to failures has been a major concern. A state-synchronized slave controller would be sufficient to recover from controller failures, but a network partition would leave half of the network brainless. In a multicontroller network, with an appropriate controller discovery mechanisms, switches can always discover a controller if one exists within their partition. Therefore, given a scalable discovery mechanism, controller failures do not pose a challenge to SDN scalability. Let us decompose the process of repairing a broken link or switch to see how it is different from the traditional networks. As shown in Fig. 3, convergence in response to a link failure has five steps. The switch detects a change. Then the switch notifies the controller. Upon notification, the control program computes the repair actions

and pushes updates to the affected data path elements, which, in turn, update their forwarding tables.2 In traditional networks, link failure notifications are flooded across the network, whereas with SDN, this information is sent directly to a controller. Therefore, the information propagation delay in SDN is no worse than in traditional networks. Also, as an advantage for SDN, the computation is carried out on more capable controller machines as opposed to weak management CPUs of all switches, regardless of whether they are affected by the failure or not. Note that the above argument was built on the implicit assumption that the failed switch or link does not affect the switch-controller communication channel. The control network itself needs to be repaired first if a failed link or switch was part of it. In that case, if the control network — built with traditional network gear — is running an IGP, the IGP needs to converge first before switches can communicate with the controller to repair the data network. In this corner case, therefore, convergence may be slower than in traditional networks. If this proves to be a problem, the network operator should deploy an out-of band control network to alleviate this issue. Overall, the failure recovery process in SDN is no worse than in traditional networks. Consequently, similar scalability concerns exist, and the same techniques used to minimize downtime in traditional networks are applicable to SDN. For instance, SDN design can and should also leverage local fast failover mechanisms available in switches to transiently forward traffic toward preprogrammed backup paths while a failure is being addressed.

3.2 Security:

IT infrastructure is rapidly moving to the cloud, creating a dramatic technology shift in the data center. This shift has significantly influenced user behavior: end users now expect anytime, anywhere access to all their data. Additionally, network operations are being transformed from operator-intensive management towards greater automation. The data center of the future is emerging as a highly virtualized environment that must address a diverse set of user needs, including anytime, anywhere access to their data, the consumerization of IT (BYOD) and increased reliance on cloud services. Security concerns are consistently identified as a major barrier to this data center transformation. While protecting user data is of paramount importance, mobility and virtualization pose new threats that must be understood and secured. And the human factor continues to lead to unnecessary downtime, expense, and unauthorized intrusion. Throughout the enterprise, end devices and data center resources including hypervisors, storage devices, servers, switches, and routers must be secured. Despite the diverse threats, existing security strategies can be successful at minimizing many of the security risks in the data center (see Figure 2). Currently available security solutions are, however, difficult to deploy, manage, program, scale, and secure. Policies are tightly coupled to physical resources as opposed to services and applications. Security solutions struggle to provide quick and automated threat mitigation across equipment from multiple vendors. Consistent security policies are difficult to administer across compute, storage, and network domains, and multiple data centers. No solutions today allow for complete security orchestration across data center networks.Today's security solutions include:

• Firewalls for perimeter defense and internal domain control.

• Intrusion detection and prevention systems that monitor network activities for malicious activities or policy violations and attempt to prevent attacks.

• Secure Sockets Layer virtual private networks (SSL VPNs), which provide the ability to securely separate customers and domains.

• Network management solutions that attempt to centrally manage many of these security functions via a console.

• IEEE 802.1X port-based network authentication and access control.

• IPsec for end-to-end authentication and encryption of the IP packets in a communication session.

• Transport Layer Security (TLS) for Application Layer communication encryption security at the Transport Layer.

• The Remote Access Dial In User Service (RADIUS) networking protocol, which offers centralized authentication, authorization, and accounting (AAA) management for end devices to use a network service.

OpenFlow-based SDN offers a number of attributes that are particularly well suited for implementing a highly secure and manageable environment:

• The flow paradigm is ideal for security processing because it offers an endto-end, service-oriented connectivity model that is not bound by traditional routing constraints.

• Logically centralized control allows for effective performance and threat monitoring across the entire network.

• Granular policy management can be based on application, service, organization, and geographical criteria rather than physical configuration.

• Resource-based security policies enable consolidated management of diverse devices with various threat risks, from highly secure firewalls and security appliances to access devices.

• Dynamic and flexible adjustment of security policy is provided under programmatic control.

• Flexible path management achieves rapid containment and isolation of intrusions without impacting other network users. By blending historical and real-time network state and performance data, SDN facilitates intelligent decision-making, achieving flexibility, operational simplicity, and improved security across a common infrastructure.

While the SDN centralized control model offers significant benefits to the network and to security management, there are tradeoffs. Logically centralized (and typically physically distributed) SDN controllers are potentially subject to a different set of risks and threats compared to conventional network architectures.

- The centralized controller emerges as a potential single point of attack and failure that must be protected from threats.
- The southbound interface between the controller and underlying networking devices (that is, OpenFlow), is vulnerable to threats that could degrade the availability, performance, and integrity of the network. OpenFlow specifies the use of TLS or UDP/DTLS, either of which supports authentication using certificates and encryption to secure the connection. Additional security measures may be needed in case this authentication fails.
- The underlying network infrastructure must be capable of enduring occasional periods where the SDN controller is unavailable, yet ensure that any new flows will be synchronized once the devices resume communications with the controller.

4. WIRELESS SDN

The current researches on SDN are mainly focused on wired network and data center, while software-defined wireless sensor network (WSN) is put forth in a few researches, but only at stage of putting forth models and concepts.

In wireless sensor network, each node may act as data source & target node, and forwarding node as well. The high dynamic characteristics of wireless link cause poor quality and low stability for link, which poses a challenge to throughput and transmission reliability of wireless sensor network. Otherwise, restricted energy and mobility requirements of node also bring difficulties to design and optimization of routing protocol.

Traditional multi-hop wireless routing is divided into active routing and passive routing; active routing such as OLSR is based on broadcast information; in each node, the routing information from that node to all other nodes is saved, so there is so much routing information that requires to be saved in each node, and too much internal storage is occupied; therefore, active routing is not adapted to high dynamic network. As for passive routing such as AODV, the routing is searched with broadcast each time when sending data is required by node; when

multiple node require sending routing, nodes need to broadcast for many times to search for routing. When there are too many links to a node, too much energy is wasted with broadcast.SDN separates control from data, and open uniform interface (such as OpenFlow) is adopted for interaction. Control layer is responsible for programming to manage & collocate network, to deploy new protocols, and etc. Through centralized control of SDN, uniform network-wide view may be obtained, and dynamic allocation may be conducted to network resources as per changes in network flow. Currently, the most routing researches for software-defined network are with respect to wired network and data center; though software-defined Internet of Things and software-defined wireless sensor network are put forth in a few researches, but only at stage of putting forth models and concepts.

In researches on SDN based on wireless network, the characteristics of wireless network, such as broadcast characteristics, hidden terminal, node mobility and etc. shall be taken into consideration. OpenFlow Protocol is only applicable to route selection, however, applying more functions such as perceiving a variety of sensor data, sleep, aperiodic data collection and etc. in wireless network node, cannot be realized with OpenFlow Protocol and Standard.

4.1 REQUIREMENTS

Implementations of the SDN solutions for traditional wired networks have considered velocity as the major performance measure. Indeed, it is necessary to guarantee that SDN nodes can execute switching operations at line rate. Satisfaction of such necessity has been paid in terms of low flexibility in the definition of the rules specifying the flows, like it will be further discussed later. In LR-WPAN networking scenarios constraints about the velocity can be relaxed. In fact, communications in LRWPANs occur at low rate by definition. On the contrary it is extremely important to guarantee low energy consumption. By looking at the scientific literature it becomes clear that reduction of energy consumption can be achieved in several ways. Among them, SDWN uses duty cycles, in network data aggregation, and flexible definition of rules and actions to allow cross-layer optimization.

Accordingly, the new requirements which have been considered in the design of SDWN are given below:

SDWN must support duty cycles– The most obvious way for reducing the energy consumption is turning the radio off when it is not utilized, that is, to use duty cycles. The radio interface of

generic nodes is turned off periodically. This would result in topology modifications which should be considered by the modules that are responsible for network Control. Obviously control of the duty cycle requires appropriate primitives.

SDWN must support in-network data aggregation– Energy consumption can be reduced by removing the from the data circulating in the network, that is, by using data aggregation techniques. In fact, it is well known that in many relevant scenarios data circulating in the network is highly correlated both in time and space. Support of data aggregation functionalities is achieved by SDWN through an appropriate module in the protocol architecture and the definition of a new action.

SDWN must support flexible definition of the rules – OpenFlow supports the definition of rules which consider the traditional TCP (or UDP)/IP header fields only. This allows a definition of switching and routing strategies which are much more flexible when compared to traditional switching/routing strategies. However, analysis of the literature suggests that higher flexibility is required in wireless sensor and actor networks. In SDWN higher flexibility is achieved along two orthogonal dimensions. In fact, on the one hand SDWN allows to define rules which consider any byte in the packet (obviously there are some limitations like we will explain in Section V-B) for matching purposes. This would allow, for example, to route packets based on the specific value in the payload they carry (which would be extremely useful in several wireless sensor and actor networking scenarios). On the other hand, SDWN allows to define rules in which matching must not be necessarily conditioned by the equality between the bytes to be considered in the packet and some reference values. In fact, SDWN allows to define rules in which matching can be conditioned by other relational operators. For example, a route can be configured for packets in which the value contained in the payload is higher than a given threshold while another route is configured for packets in which the above value is below the threshold. Again several application scenarios can be figured out in which the above flexibility is extremely useful. Other requirements– There are several other obvious new requirements that must be satisfied, which we will not specifically analyze for space constraints. Examples include the necessity to support nodes mobility and the resulting topology changes, the necessity to deal with the unreliability characterizing wireless links, and the need to be robust to the failure of generic nodes and the Control node.

4.2 PROTOCOL ARCHITECTURE

The protocol architecture Developed for a sensor and actor network based on IEEE 802.15.4 communication nodes. Besides the transceiver a micro-control unit is deployed in such nodes with limited computing and energy capabilities. In the network there is also one (or several) sink node(s) which is (are) also the node(s) where the network Controller is executed. The IEEE 802.15.4 transceiver of the sink is connected to an embedded system running, for example, a Linux operating system. Computing/communication capabilities of such embedded system are

significantly high when compared to the other nodes of the network. The network Controller functionality will be performed by such embedded system. In Figure 1 we show the proposed protocol architectures for SDWN nodes. More specifically, in Figure 1 (left) we show the protocol architecture for generic nodes, whereas in Figure 1 (right) we show the protocol architecture for the sink nodes.

A. Generic node

All generic nodes run the basic physical and MAC layer functionalities of standard IEEE 802.15.4, required to form a peer-to-peer topology. On top of the MAC layer, a Forwarding layer is executed which is responsible for treating arriving packets as specified by the controller. To achieve such goal, the Forwarding layer maintains a flow table updated. According to the SDN approach, the entries of the flow tables, which are called, are defined by a rule, an action, and statistic information.

A rule is a description of the characteristics which are featured by packets belonging to a flow and that must be treated by the node in the same way. Indeed, each flow table entry specifies the action which should be executed to all packets satisfying the above rule. Finally, the table flow entry specifies the number of received packets which have satisfied the rule, that is, the statistic information.

Arriving packets are provided by the MAC layer to the Forwarding layer. This identifies the type of packet and if it is a control packet, then sends it to the Network Operating System layer which will be described later in this section. Otherwise, i.e., if the packet is a data packet, the Forwarding layer controls whether the packet matches one of the rules specified in the flow table. If this is the case, then the packet is treated according to the corresponding action. Otherwise the packet is given to the Network Operating System layer. The Aggregation layer is executed over the Forwarding layer. Its responsibility is to perform the action required to aggregate information flowing through the network. Current implementation of the Aggregation layer is quite straightforward. In fact, one of the actions which can be specified by the flow table entries is to include the packet in an aggregation equivalent flow (AEF). Packets belonging to the same AEF can be aggregated with each other and sent to the Aggregation layer for this purpose. The Forwarding layer will just concatenate arriving packets, if these are sufficiently small, and forward them as specified by the corresponding flow table entry.

In the future, a more sophisticated behavior for the Aggregation layer can be designed. Finally, in the architecture shown in Figure 1 (left) we can distinguish a Network Operating System (NOS) layer which runs on top of the IEEE 802.15.4 standard physical and link layers and has access to all the new protocol layers described.

Above. Indeed, It is observed that the NOS layer has access to APIs offered by all layers. Such APIs enable to control the behavior of the physical and link protocol layers as well as the new defined layers and therefore, allow cross-layer operations.

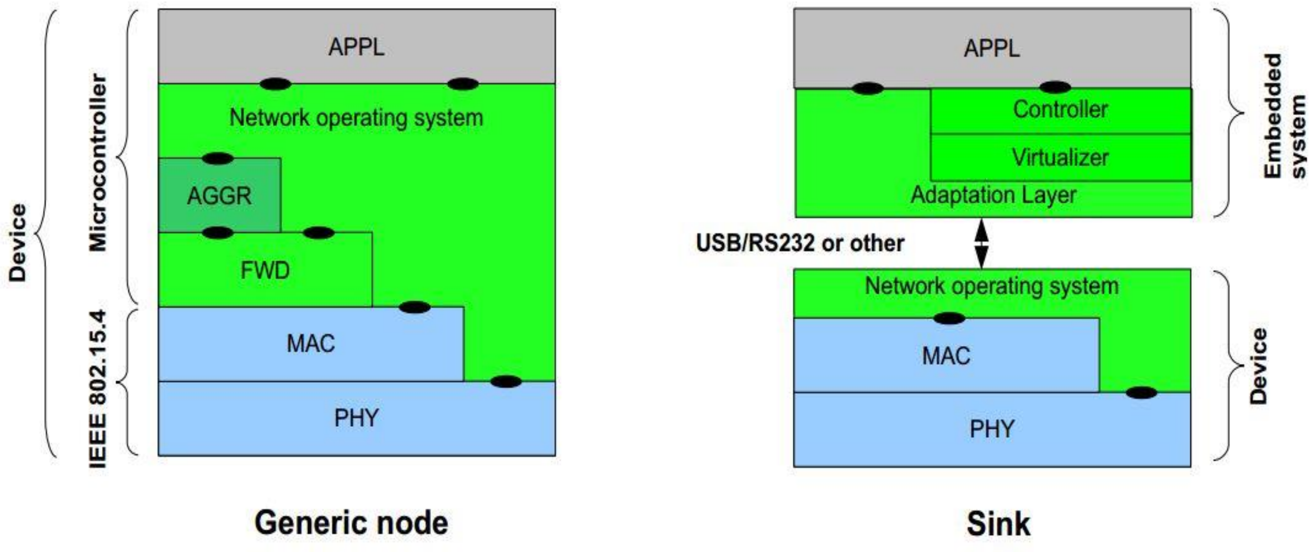
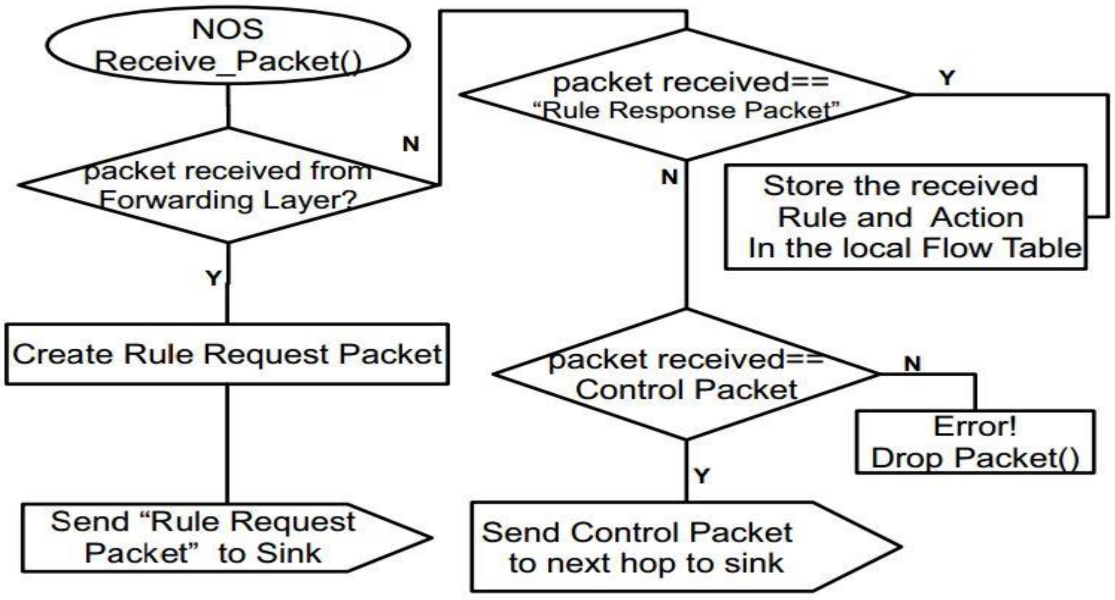


Figure: Node architecture

Objectives of the NOS layer are, on the one hand, to collect local information from the node and send such information to the Controller; on the other hand, to control the behavior of all other layers of the protocol stack as specified by the Controller. Achievement of the above objectives requires NOS to be able to reach the sink node at any time. To this purpose a simple protocol is applied as described in Section V-A. A Flow Chart of the operations run by the NOS layer is reported in Figure:



NOS Receive_Packet()
packet received from Forwarding Layer?
N
Y
Create Rule Request Packet
Send "Rule Request Packet" to Sink
packet received== "Rule Response Packet"
Y
Store the received Rule and Action In the local Flow Table
N
packet received== Control Packet
N
Error! Drop Packet()
Y
Send Control Packet to next hop to sink

Figure: flow chart for NOS operation

Finally, note that the Application layer runs on top of the NOS layer, which is thus expected to provide an appropriate API. In order to support legacy applications, the current API generalizes the IEEE 802 APIs.

B. Sink node

The architecture of the Sink node can be split into two different parts as shown in Figure 1 (right). In fact, the bottom layers which run in the same device which is used by generic nodes are the same as explained in the previous subsection. Besides, there are further functionality which require high computing and communication capabilities and therefore are executed in the Linux-based embedded system.

The device and the embedded system are usually connected through USB, RS232 or some other communication interface. In the embedded system, an Adaptation layer is executed which is responsible of formatting the messages in such a way that they can be handled by the WPAN devices.

Another key element of the sink node is the **Virtualizer layer**. This layer uses the local information collected by the generic nodes to build a consistent and detailed representation of the current network topology (graph, energy level of all generic nodes, quality of the links, etc). The Virtualizer layer is responsible for allowing the coexistence of different logic networks on the same devices. Such networks can use different policies regarding the network management, i.e. they use different Controllers. Each coexisting network is characterized by a network rule which is used to filter packets that will be treated according to the specific management policies defined by the corresponding Controller. For what concerns the representation of the network topology, a map is used to collect all the information regarding a generic node. More specifically, for each generic node A the topology information that we need to manage is the following:

• Last time instants when the sink node has received a packet generated by A.

• The battery level reported in the last packets received from A.

• A list of nodes that are neighbors of A. For each of such neighbors, say B, we need to represent the address (at this moment this is a 2 byte field), the quality of the link between nodes A and B, a time stamp reporting the last time instant when the sink has received a packet from A that reports B among A's neighbors.

In current implementation we represent the network conditions by exploiting the Java Map interfaces. For the Map an obsolescence timeout is defined which is the time after which an entry (being it an entire list of neighbors or just one neighbor) should be removed from the map unless there are evidences that this action should not be taken.

Finally, besides the Application layer the protocol architecture of the sink node contains one or several Controller(s). The Controller is responsible to implement the desired network management policy. More specifically, the Controller will receive packets that generic nodes

have not been able to classify in an existing flow and for such packets must define a rule along with the appropriate actions. The Controller will create flow table entries in this way which are based on the information on the current topology of the network.

4.3 DESIGN AND IMPLEMENTATION:

Collection of topology information

- The sink node periodically generates a *beacon packet*.
- This packet is broadcasted throughout the network in multi-hop
- At each hop the beacon packet contains information about the current distance from the sink (expressed as the number of hops to reach it) and a measurement of the local battery level.
- Upon receiving a beacon packet each node increases the value of the current distance from the sink by one, overwrites the value of the current level of the battery, and, then, forwards the packet.
- Also, upon receiving a beacon packet each node measures the RSSI value in the link towards the nodes that have just transmitted the beacon
- The information contained in the beacon packets and the RSSI measurement allow each node to identify the most convenient next hop node to reach the sink.
- Each node also stores in a local table, called *neighbors table*, the list of nodes from which it received a beacon, and for each of them stores the measured value of RSSI as well.
- This list of neighbors will be sent periodically to the sink node using a packet, called *report packet*, which will be forwarded to the best next hop node
- The sink node will receive the list of neighbors from any network node and will be able to infer the global topology of the network

Packet format:

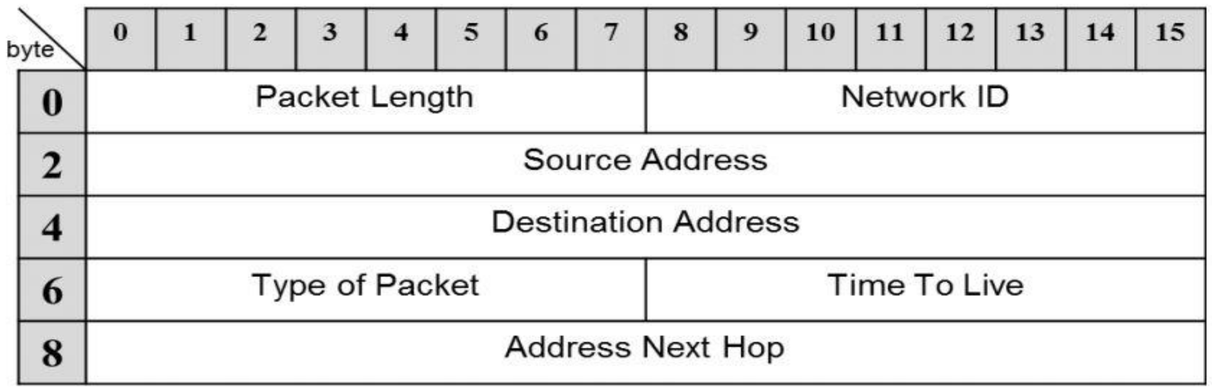


Figure: packet format

All the packets circulating in the network use a fixed header that is organized as depicted in Figure. Additional fields may be included based on the specific type of packet, as explained below:

Type 0 - Data packet: This packet is generated/delivered by/to the application layer and contains the header as depicted in Figure plus the payload.

Type 1 - Beacon packet: This is the packet which is broadcast periodically by the sink. Such packet contains the header fields as reported above and an additional byte (referred to as "Hops to sink") providing the number of hops required to reach the sink from the node which has transmitted (not generated) the packet.

Type 2 - Report packet: This packet is generated periodically by each node and is transmitted to the sink upon receiving a Beacon Packet. Report packets contain the 10 bytes of header depicted in Figure plus the "Hop to sink" byte, another byte reporting an assessment of the current charge level of the battery (several devices support the reading of the battery voltage) and information about the current neighboring nodes. Such information is structured as follows. The first byte contains the number of current neighbors. Then for each of the above neighbors 2 bytes are used for the address and an additional byte for the RSSI.

Type 3 - Rule/action Request: This packet is generated by a node upon receiving a packet which does not match any of the rules stored in the flow table. It is generated as a copy of the incoming packet in which, however, the Type of Packet field (i.e., byte 6 in the header) is set to 3 and the original Type of Packet value is inserted at byte 10.

Type 4 - Rule/action Response: Besides the usual header this packet contains a pair Rule/Action which is described as presented in Section V-B. More specifically, we use 2 bits to identify the window size, 3 bits to identify the relational operator, one byte to identify the position in the packet of the first byte of the window, and two bytes for the "value" field

5. BROADCAST STORM

A straight-forward approach to perform broadcast is by flooding. A host, on receiving a broadcast message for the first time, has the obligation to rebroadcast the message. Clearly, this costs n transmissions in a network of it hosts. In a CSMA/CA network, drawbacks of flooding include:

Redundant rebroadcasts: When a mobile host decides to rebroadcast a broadcast message to its neighbors, all its neighbors already have the message.
Contention: After a mobile host broadcasts a message, if many of its neighbors decide to rebroadcast the message, these transmissions (which are all from nearby hosts) may severely contend with each other.

Collision: Because of the deficiency of backoff mechanism, the lack of RTS/CTS dialogue, and the absence of CD, collisions are more likely to occur and cause more damage. Collectively, we refer to the above phenomena as the broadcast storm problem

Simply flooding broadcast package in network would cause problems such as rebroadcast & redundancy, signal collision, broadcast storm and etc. Especially when network nodes are relatively dense, these problems would be more outstanding. Generally, wireless sensor network is deployed densely, and there are a lot of redundant nodes, and system bears stronger fault-tolerant performance. If only a part of nodes are selected for rebroadcast on premise that all nodes should receive broadcast, the problem of broadcast storm would be relieved.

5.1 Related works:

At present, there are a variety of researches that aim to solve the problem of broadcast storm, there into, there are algorithms based on probability, counter, distance, location, neighbor information and etc.

Probabilistic Scheme: An intuitive way to reduce rebroadcasts is to use probabilistic rebroadcasting. On receiving a broadcast message for the first time, a host will rebroadcast it with probability P. Clearly, when P = 1, this scheme is equivalent to flooding.

Counter-Based Scheme: When a host tries to rebroadcast a message, the rebroadcast message may be blocked by busy medium, backoff procedure, and other queued messages. There is a chance for the host to hear the same message again and again from other rebroadcasting hosts before the host actually starts transmitting the message. We can prevent a host from rebroadcasting when the expected additional coverage of the host’s rebroadcast becomes too low. This is what the counter-based scheme is based on. Specifically, a counter c is used to keep track of the number of times the broadcast message is received. A counter threshold C is chosen. Whenever c 2 C, the rebroadcast is inhibited. The scheme is formally derived below.

S1. Initialize counter c = 1 when a broadcast message is heard for the first time. In S2, if message is heard again, interrupt the waiting and perform S4.

S2. Wait for a random number of slots. Then submit message for transmission and wait until the transmission actually starts.

S3. The message is on the air. The procedure exits.

S4. Increase c by one. If c < C, resume the interrupted waiting in S2. Otherwise c = C, proceed to SS. Cancel the transmission of message if it was submitted in S2. The host is prohibited from rebroadcasting msg. Then exits.

Neighbor information based scheme: the algorithm where MPR nodes are selected by OLSR routing is taken into reference; the neighbors of a part of nodes are selected for broadcast. 1-hop and 2-hop neighbor nodes of some node are utilized in this algorithm..

As for probability based method, nodes conduct broadcast based on certain probability; however, this method could not be adapted to change in node density, if the node density is low, the area covered by broadcast decreases. As for counter-based algorithm, after the number of broadcast received by a node exceeds a certain threshold, the broadcast at that node would be canceled. This algorithm is not influenced by node density in network, but there is much broadcast delay. As for broadcast algorithm based on neighbor information, a part of nodes are selected for broadcast as per neighbor information. This kind of broadcast algorithm needs neighbor information.

Tests were conducted for 4 algorithms (3 broadcast methods and full-node broadcast) in simulation scene; the results of performances contrast are shown in Table.

Method	Number of Broadcast	Lifetime(s)	Parameter
All nodes	800	744	
Probability	233	773	p=0.3
Counter	201	784	threshold=3,waittime=0.02s
Greedy Neighbor	159	798	

There are 800 nodes in total in simulation network, the number of nodes in full-node broadcast is the number of total nodes, while the number of broadcast in the other 3 methods is largely reduced, there into, the number in counter method is more than that in greedy neighbor method but less than that in probability method. It can be seen that the less the number of broadcast is, the longer the network lifetime is. What should be noticed is that as for probability method and counter method, if different parameters are set up, the results are different; if the probability set up in probability method is larger, or if the threshold set up in counter method is larger, the number of broadcast is larger.

The parameters for probability method and counter method in the table are values with better performance in experiment. During actual simulation, even greedy neighbor algorithm has multiple redundancies, because overlap exists for greedy neighbor of multiple nodes in transmission distance after multiple hops, and there is still margin for reduction.

Node forms the backward path to controller as per broadcast package received, and sends REPORT packet along the backward path; if the information of each node is sent separately along the backward path, then midway node could finish sending information of downstream node through sending for many times. In this paper, it is designed that the upstream node shall combine information of all next-hop nodes for sending, after information of downstream node arrives at upstream node.

After a node receives SDN broadcast package, there is certain delay before it sends REPORT packet; it is designed that the delay time of node is inversely proportional to hop count of the node to controller. The larger the hop count is, the shorter the delay for sending node information package is. Therefore, the information of nodes located at the edge would be reported firstly, and

After controller receives REPORT packet, node information shall be saved into array of node information list, and residual energy of node shall be saved into array of residual energy. Thus there is global view at controller, and controller is able to provide routing for other nodes.

5.2 Existing problem:

From the above discussion we can see that to reduce the broadcast storm we can use probabilistic broadcasting or the counter based broadcasting to reduce the total number of broadcast. But in both of the cases we have some shortcomings. They are:

- For probability based method, nodes conduct broadcast based on certain probability; however, this method could not be adapted to change in node density, if the node density is low, the area covered by broadcast decreases.
- For counter-based algorithm, after the number of broadcast received by a node exceeds a certain threshold, the broadcast at that node would be canceled. This algorithm is not influenced by node density in network, but there is much broadcast delay.

5.3 Probable Solutions:

To reduce the broadcast storm there are some more scheme. These schemes can be applied to simulate and see the result on broadcast storm. Some of the scheme are described below:

Distance-Based Scheme: In the previous scheme, a counter is used to decide whether to drop a rebroadcast or not. In this scheme, we will use the relative distance between hosts to make the decision.

Location-Based Scheme: Without loss of generality, let a host's location be (0,O) (here we use q-coordinate to facilitate our presentation; in fact, devices such as GPS receivers can provide 3-D locations in longitude, latitude, and altitude). Suppose a host has received the same broadcast message from k hosts located at (x1,yl),(x2,y2)... , (xk,yk). We can calculate the additional area that can be covered if the host rebroadcasts the message. Let AC (x1,yl),(x2,y2)... , (xk,yk) denote the additional coverage divided by r^2. Then we can compare this value to a predefined coverage threshold A ($0 < A < 0.61$) to determine whether the receiving host should rebroadcast or not.

Cluster-Based Scheme: It is assumed that a host periodically sends packets to advertise its presence. Thus any host can determine its connectivity with other hosts on its own. Each host has a unique ID. A cluster is a set of hosts formed as follows. A host with a local minimal ID will elect itself as a cluster head. All surrounding hosts of a head are members of the cluster identified by the head's ID. Within a cluster, a member that can communicate with a host in another cluster is a gateway. To take mobility into account, when two heads meet, the one with a larger ID gives up its head role.

S1. When the broadcast message is heard, if the host is a non-gateway member, the rebroadcast is inhibited and the procedure exits. Otherwise, the host is either a head or a gateway. Proceed to S2.

S2. Use any of the probabilistic, counter-based, distance based, and location-based schemes to determine whether to rebroadcast or not.

Shortest path based scheme:

- Implement shortest distance based broadcasting:
    - At the very first the network topology will be discovered by counter based scheme
    - Then from the topology the shortest path for each of the node from controller is calculated
    - The maximum distant node from the controller is taken as threshold matric and is set
    - After each period the controller will broadcast the beacon packet with the metric 0.
    - After receiving a packet if the metric is greater than the threshold then will be discarded otherwise broadcasted.
    - If any new node is added in the network it will send join request to all available nodes and node will send the information to the controller. The controller will choose the shortest path for that new node.

WSN DESIGN ISSUES

Energy:

Sensor nodes run on limited battery, and in most cases regular maintenance is not possible. So, it is vitally important that energy usage is wise and efficient so it prolongs the network lifetime. Some cases may allow rechargeable batteries and energy can be replenished through solar or wind power. However, as SDN grows it might get difficult to do so.

Communication:

IEEE 802.15.1 and 802.15.4 are the two most viable protocols for WSN. These protocols co-exist with other protocols which are not ideally suited for WSN, such as WLAN and UWB.

Bluetooth(802.15.1) is a short range wireless communication technology which consumed high amount of energy until BLE (Bluetooth low energy) was available. BLE reduces the pwer consumption by a lrage amount and consequently increases communication range. The only drawback to Bluetooth is with regards to scalability.

IEEE 802.15.4 was designed specifically for networks with low power consumption, low deployment cost, less complexity and short range communication while maintaining a simple protocol stack. The MAC layer defines two type of functional nodes namely: Reduced Functional nodes and Full Functional nodes. Reduced functional nodes can only act as an end device but full functional nodes can also act as the network coordinator; can be the controller in SDN paradigm.

Routing:

WSN topologies are unstructured and therefore many traditional routing protocols are not suitable. The routing protocols should be lightweight because of the limited resources. WSN protocols are classified as greedy forwarding, data centric, energy centric and flood based.

Greedy forwarding forward the packets to neighbors closest to the destination, which can be effective in a dense deployment of nodes. Data centric approaches are attribute based and help removing redundant data. Compression and aggregation in data centric approaches also minimize energy wastage for redundant data. Energy efficient protocols choose neighbors with high energy levels and route packets. Energy aware harvesting protocol can be applied for efficient use of harvested energy from the environment, integration of EH-WSN into SDN is discussed later.

QOS:

Coupled with configuration management is performance management. The two main objectives of performance management of a WSN are the Quality of Service (QoS) and the quality of the information obtained . However, there is usually a trade-off between QoS, energy consumption and network lifetime. Depending on the environmental conditions, resource constraints, service demand; the management protocols can permit lower QoS when energy is scarce and thus extending network lifetime. Several network management protocols have been proposed in QoS management of the WSN architecture including the hybrid data dissemination framework protocol RRP, Sensor Network Management System (SNMS), Simple Network Management Protocol (SNMP) and Wireless sensor network Management System (WinMS) . SNMP is a

widely used management protocol that provides good management for TCP/IP based networks and is supported by several vendors in managing both wired and wireless networks. It allows management of network performance and enables fault identification. WSNManagement based on SNMP protocols has been proposed as an efficient network configuration, fault and performance management system. The performance of WSNManagement showed efficiency in WSN management with a 5 percent reduced packet loss and a 0.2 s reduced delay time while improving the lifetime of the overall sensor node system. Delay-related QoS is critical for WSNs, especially for real-time applications. Apart from WSNManagement, Zhao et al. propose an optimized resource allocation solution for delay constrained Wireless Regional Area Networks (WRANs) to improve delay-related QoS management.

Security:

Management of security is difficult to provide as WSNs are mainly made up of ad-hoc wireless networks with intermittent connectivity and resource limitations. This leads to vulnerability to threats that could be internal, external, malicious or even accidental. Data and resources can thus be stolen or modified and a denial of service attack is also possible. Security management for example has been approached through the use of key management schemes. Reegan et al. mention various requirements in managing security such as authentication, confidentiality, integrity, availability and authorisation. A network with high security features should be developed to meet these requirements however, the challenge here is the limited resource in bandwidth and energy inherent to WSNs . Management of security therefore has to be based on effective key distribution taking into account features such as bandwidth, sensor memory, transmission range and the necessary know-how . Several key management schemes have also been discussed by Reegan et al.. Much more recently Key management based on the dynamic clustering and optimal routing choice of the Mobile Sink has been proposed . The scheme extends static key management to dynamic key management with an improved flexibility while satisfying storage efficiency and connectivity. A further contribution has been made by proposing a hybrid key management scheme which takes advantage of both the Polynomial Pool and Basic Random (PPBR) key pre-distribution techniques to improve the difficulty in cracking the key system . To tackle issues related with link connectivity in key management, the tree base path key establishment method is used. The architecture of WSNs based on SDN is such that a logically centralized controller poses a security risk as any compromise in the controller, cluster-head in clustered topologies or even the link to the controller can affect the system QoS. A malicious attacker can introduce false data and parameters in the network or a Denial of Service resulting in system unreliability. A few contributions have been made in managing security for SDN-based WSNs, Flauzac et al. for example describe the need for security management in both wired and wireless networks based on SDN for Internet of Things (IoT) and propose a security model for IoT SDN architecture. An extension of the proposed architecture for IoT is made to

include sensors integrating the aspect of guaranteeing the security of the network based on the grid of security concept embedded in each controller. The grid of security concept has been presented previously as an approach to securing networks by ensuring that communication takes place between trustworthy devices regardless of system policy control supported by a communication model to help structure the service distribution of security among nodes. The overall concept is to decentralize the management of security of the network to ensure continuity in security during failure.

Another security management contribution has been the integration of Secure Multiparty Computation (SMC) to secure private sensory data in SDN-based WSNs. The focus is the application of SMC in securely processing sensory data between the SDN-based WSN and the web server. Sun et al. present a security model based on the SMC protocol allowing clients in the SDN-based WSN to connect and disconnect to the web server arbitrarily. A lottery SMC protocol is constructed for the performing selection in SDN-based WSNs with an encryption based on the layered homophobic function.

Scalability:

The architectures reviewed so far are based on a centralized controller in a flat topology which is easy to deploy and manage. Gante et al.for instance introduce smart management of SDWSNs to improve efficiency and overcome inherent difficulties of ordinary WSNs. The management scheme is based on a proposed base station architecture for WSNs with an integrated controller. The controller creates forwarding rules that are placed in flow tables from location data obtained through localization techniques processed in the application layer of the architecture. However, flat topologies like this are limited to short range and small scale applications and as networks scale up to very large numbers of nodes there is need to integrate topologies that are hierarchical or provide a virtual overlay of the physical network to support efficient scalability and localization. A few architectures are available for network management designed towards improving SDN-based WSN management efficiency by using distributed multiple controllers rather than a single centralized controller. This also enables efficient scalability and a reduction in overhead control data as it is not necessary to communicate all control data centrally. Distributed controllers also allow for effective security and QoS management as the WSN is not solely dependent on one controller, making the system more reliable during attacks or failures in the link to the central controller.

A hierarchical architecture called Software Defined Clustered Sensor Networks (SDCSN) has been proposed to use multiple base stations as controllers that also play the role of cluster heads. A large-scale group of nodes is divided into clusters and there is a cluster head for each. The cluster head controls and coordinates the sensor nodes in each cluster and all the information processed in each cluster is routed to the cluster head. Management of such an architecture requires enabling the sensor nodes to function as controllers effectively enabling multiple

controllers in the network. Oliveira et al. design and implement an architecture based on multiple controllers within a WSN in a framework called TinySDN based on Tiny-OS with a design structure that consists of SDN-enabled sensor nodes and an SDN controller node. This addresses issues such as in-band control, higher communication latency, and limited energy supply. The downside to this management approach is that the cluster head can also become vulnerable to attack posing a network security risk however self-stabilisation techniques for re-selecting a new cluster head upon such an event have been proposed. To manage the network with more flexibility for integration of various functions an architecture based on network virtualisation has been proposed.

Another hierarchical architecture proposed to manage scalability is context based logical clustering . This is based on logically clustering sensors according to context (gathered data type) regardless of their physical distribution based on HyperFlow unlike other clustering methods proposed which allow clustering of adjacent nodes. Clustering sensors according to context would allow for data and resource sharing regardless of network expansion. Each cluster has a local controller which maybe physically distributed but logically synchronised hence referred to as logical sink in the virtual network overlay. The performance study shows that this technique improves network stability effectively.

Sensor data without location information is not useful especially in large-scale applications with hundreds to thousands of nodes. SDN in WSNs provides the possibility of obtaining location information with a good level of accuracy by implementing proposed localization algorithms .The localization algorithm to be implemented and managed can either be centralized or distributed depending on the mapping information and level of accuracy provided by the algorithm. More recently a localization node selection algorithm based on the SDN and the Cramer-Rao Lower Bound (CRLB) metric has been investigated and proposed . The algorithm makes use of the global network view that the SDN controller has while satisfying the need for energy conservation and the simulation results presented a significant improvement in the localization performance.

Design of a sensor openflow:

The fundamental assumption that OpenFlow makes is that the underlying network is composed of high-speed switches such as Ethernet/MPLS switches and IP routers. OpenFlow was also designed as a wired protocol, and its typical use cases are data centers, cellular backhaul, enterprise WiFi backbone, and WANs. Compared to WSN, these represent significant disparities and lead to major challenges in designing SOF. A. Data Plane: Creating Flows At the data plane, packets are handled as per flows. A flow is a programmable (i.e., user-customizable) set of packets that share certain properties specified by a Match in a flow-table entry, or flow entry for short. The Match is often wildcarded, like "IP source address is 10.0.*.*", and packets that match it will be treated as in the same flow and be imposed an Action (e.g., "send to port 1") specified by the same flow entry (cf. Fig. 1 or 2). OpenFlow implicitly assumes the presence of IP-like

addressing so as to create flows. For example, it defines Match fields such as IPv4_SRC, ETH_DST and MPLS_LABEL to match packets with IPv4 source address, Ethernet destination address and MPLS label, respectively. However, as opposed to address-centric OpenFlow networks, WSN are typicallydata-centric—acquiring data of interest is more important than knowing who sent the data—and employ different addressing such as attribute-based naming, e.g., "nodes whose sensing temperature>30◦C". SOF must cope with this in the first place for flow creation. B. Control Plane: SOF Channel An OpenFlow channel is an end-to-end connection used to transmit control messages between a controller and a switch. An SOF channel is similarly defined. However, this channel must provide TCP/IP connectivity for reliable end-to end in-order message delivery, and the two end-parties are identified using IP addresses.. This is normally not practical for WSN and the SOF channel has to be hosted in band, i.e., by the WSN itself. Thus, the resource constrained WSN will have to additionally carry control traffic between controllers and sensors. This is further exacerbated by the fact that control traffic in WSN tends to be large due to the high network dynamics (node/link failures, energy-aware routing, mobility, etc.). Hence, without a proper mechanism to curb control traffic overhead, the underlying WSN can be overloaded. D. Traffic Generation End-users are considered peripheral to SDN and hence out of the scope of OpenFlow. On the contrary, sensor nodes behave like end-users by generating data packets, in addition to merely forwarding data as OpenFlow switches do. E. In-Network Processing At times, WSN need to process data in-situ, e.g., perform data aggregation or decision fusion, in order to reduce data redundancy and conserve network resource such as bandwidth and energy. This is another feature absent in SDN. F. Backward and Peer Compatibility Albeit radical, SOF should desirably provide backward compatibility, with respect to traditional (non-OpenFlow and non-SOF) networks, so as to protect legacy investments. It is also desirable for SOF to offer peer

compatibility, i.e., to be compatible with OpenFlow networks which are concurrently being developed and standardized, for interoperability purposes.

7. WSN APPLICATION AND MOTIVATION FOR SD-WSN

Wireless sensor networks are able to integrate various sensing capabilities thus providing support for various real-world applications. This flexibility is accompanied by several research challenges in providing effective management for the application considering the resource constrained nature of WSNs. The various benefits that SDN introduces to these management challenges allows it to be utilized in several applications such as in environmental applications, health care, military and in home networks. We shall briefly discuss these application scenarios and bring to light the benefits of SDN-based management in these areas.

Environmental Applications

A popular environmental application of WSNs is based on the measurement of physical parameters such as temperature, vibration, sound, chemicals and gases. This has led to application in scenarios such as the agriculture industry for irrigation or animal monitoring, water industry for development of smart water grids to enable efficient distribution and control of potable water and in disaster relief applications for monitoring and fighting forest fires and also in earthquake detection and rescue operations. Other applications include environmental control of pollutants and marine monitoring requiring long term unmanned WSN operations. However, such applications may require deployment of sensors over very large areas, harsh environments or even hard to reach regions which may result in dependence on sensor energy supply which affects network lifetime and the ability to alternate tasks upon change of sensing requirement or upon node failure. SDN-based management would allow for a centralized network view of essential parameters suitable for large-scale applications, where energy is crucial SDN techniques provide effective resource allocation through software duty-cycling and by enabling easier use of traffic engineering, congestion control, load balancing and mobility management. SDN also provides an effective interface for sensor re-programmability necessary for function alternation without requiring additional hardware such as FPGAs .

Medicine and Health Care

WSN application in health care has been considered to have the potential of being beneficial to the quality of health especially for the chronically ill and elderly patients. WSNs can also be used for tracking of patients and doctors in hospitals which can prove useful in saving lives. A controversial application of WSNs in health care is the use of implantable sensor systems to monitor patient activities resulting in Wireless body Area Networks (WBANs). Darwish et al. identify challenges to implementing WBANs among which are sensor maintenance once body-worn, energy management for sensor batteries considering that charging would be pose a challenge especially for the elderly as they have to remember to charge multiple sensors, insufficient bandwidth resource to allow for transmission of large amounts of medical diagnostic data, meeting the QoS requirements of health monitoring hindered by WSN resource constraints, reliability of critical medical data packets being delivered, security and privacy of medical data,

scalability and sensor mobility challenges as patients move about in their daily life. SDN-based management would provide a solution to several of these highlighted challenges based on the ability to provide energy and maintenance management from a central location which could be a task of hospital administration for instance. Furthermore, SDN techniques are being developed to reduce traffic overhead data to mitigate bandwidth problems and also to ensure accuracy of data and fault monitoring more efficiently, SDN provides a solution based on separation of destructive interfering nodes. Security and privacy is a key issue in health applications and SDN provides several encryption algorithms that can be delivered on demand, the encryption techniques available are discussed later in this paper.

Military Applications

In military applications WSNs can be used to detect presence of enemy or friendly forces, assessment of terrain and chemical sensing of weapons. Recently Fraga-Lamas et al. reviewed the possibility of applying IoT to warfare to increase efficiency. Uses of WSNs for defence IoT highlighted included application in fire control, inventory tracking, fleet monitoring, energy management and management of the health and safety of troops. However, to be suitable for use in defense and security WSNs need to meet operational requirements such as robustness, ease of deployment, interoperability/adaptability and most importantly security. SDN can be leveraged to better achieve these requirements as it maintains a global control of the network infrastructure in the field which better manages system security and issuance of new task protocols during operation. SDN-based management also provides an opportunity to easier deploy mobility algorithms to handle constant movement of troops as these algorithms can be configured and managed in a more resource capable global controller.

Wireless Home Networks

WSNs for home use have seen increasing demands with arising smart homes and buildings in general. Homes are now composed of a network of light, motion and temperature sensor nodes that can be integrated in appliances to introduce smart capabilities in a home. This network can also be linked to the Internet to allow users to remotely control these home appliances. However, in addition to a network of smart devices is a network of other communication technologies such as WiFi, ethanet, cellular and power-line communications. The challenge with such a heterogeneous network is the lack of an automated system to enable network optimization by selecting the best communication technology to use, this decision is usually made by the user manually. Soetens et al. [71] have proposed an SDN-based management technique to improve management of such heterogeneous home networks by using OpenFlow-enabled link switches. SDN techniques have also been proposed to enhance the development and management of smart homes [72]. Smart device use is increasing in homes and this creates difficulty in management especially with the advent of video streaming services such as Youtube and Netflix.

8. ENERGY CONSTRAINTS ON WSN

Recharging batteries in a wireless sensor network is sometimes impossible due to the placement of the sensor nodes, but more commonly it is merely practically and/or economically infeasible. At any rate, it is widely recognized that, generally, energy is a strictly limited resource in wireless sensor networks and that the consequences of this limitation must be considered. Ultimately, if we want to have the sensor network performing satisfactorily for as long as possible, the energy constrained operation of the sensor nodes forces us to compromise between different activities in the network. Compromises are needed on the node level as well as on the network level. Saving energy is tantamount to finding the best compromise, the best tradeoff, between different energy consuming activities and their design.
Equipping the sensor nodes with non-rechargeable batteries of appropriately chosen capacity may justify the high cost of WSN deployment for low consumption applications, but may not for power-hungry applications such as audio/video systems.

Energy-harvesting wireless sensor networks (EH-WSN) provide a solution to this energy problem by allowing sensor nodes to harness environmental energy to power the nodes. This solution can be implemented through two different design paradigms – *perpetual operation* and *supply augmentation.*

In the former, consumption is adapted to available harvest so that the node, theoretically, operates forever. In the latter, the objective is similar to that of traditional finite battery supplied WSNs, which is to maximize some energy metric such as network residual energy or lifetime. This is especially applicable to monitoring applications that require continuous operation even when environmental energy is unavailable. We adapt this scenario for our design objectives. Two enabling technologies drive the feasibility of EH-WSNs – *energy harvesting* and *energy management.* Energy harvesting technology converts environmental energy into electrical energy, for example, Photovoltaic (PV) panels from solar and piezoelectric transducers from mechanical stress.

A route selection method in which route decisions are based on a cost metric comprised of two components – the first relating to energy consumption due to transmission of packets and the second reflecting the energy harvest wastage due to overcharge can be an excellent addition to the routing policies supported by SDN.

8.1 Related works

Several works on energy-harvesting routing protocols have been published introduced one of the first routing protocols to integrate solar energy harvest into the route selection by simply classifying nodes as either harvesting or non-harvesting, with the non-harvesting nodes avoided as much as possible. Authors associate cost metrics representative of a node's available energy with the links. Routes are then found using Bellman-Ford, Directed Diffusion (DD), or other applicable shortest-path or least-cost algorithms applied on these metrics.
Routing is cast into a flow-maximization problem in which the probability of transmitting a packet over an edge is proportional to the maximum flow of that edge, calculated using an extension of Ford-Fulkerson algorithm.

A metric based on geographical distance, link quality and energy harvest is used in a blacklisting routing algorithm, however the protocol requires nodes to know the geographical location of other nodes, which may not be available. An energy budget for a time slot is calculated. During the exploratory phase of DD, only nodes with enough energy budget admit packets for relay. The protocol proposed is based on a finite state machine with charge, receive, and transmit states for a node. The node transitions to the charge state when its battery level falls below a threshold.

An energy harvesting adaptive opportunistic routing algorithm in which data packets are broadcasted towards the sink. For high data rate applications, this may produce too many duplicate data packets.
These works have shown energy savings can be obtained through harvest-aware routing protocols, however, their cost metrics do not factor in network energy wastage due to overcharge. Further performance improvement can be achieved by incorporating energy wastage in the routing decisions.

9. ENERGY HARVESTING-WSN IMPLEMENTATIONS

On-Demand Routing:

A common classification of WSN routing protocols is the categorization into *table-driven* or *on-demand*. On demand routing protocols discover routes only when required.
This can be advantageous when there is high mobility, high variability in the network energy state such as in the case of EH-WSNs, or when sources only need to originate packets at irregular events.

Table-Driven Routing:

In table-driven or proactive routing protocols, each node maintains routes to every possible destination. However, active maintenance of these routes can become inefficient when there is high mobility or when traffic is characterized by non-regular events.

Motivation for Integration to SDN

- Nodes can work as simple forwarding elements
- No on-demand routing overhead
- Control packets are decoupled from data packets
- Less space complexity in nodes, requiring only the energy level information of the neighbor nodes
- Heterogeneous nodes requiring different routing policies.

10. Probable SDN based Solution

- Each time report packet arrives at controller, estimated energy harvest value is collected by the controller.
- When a node requests the controller for a route, the controller can find a route to the destination which wastes the minimal amount of energy and convey it to the sender node.
- This eliminates the latency that reactive routing algorithms incur.
- Also, more than one routing matrices can be used in one implementation. Different policies can be applied to different nodes with different requirements.

11. Future works

- The packet format should be designed to support necessary change
- Different algorithm should tested to find the optimization to find shortest path.
- The protocol with the change should be simulated to see the result
- Different broadcasting scheme should be simulated to find the optimization.

References

1. **New Networking Era: Software Defined Networking -** *Furqan Alam, Iyad Katib, Ahmed Saeed Alzahrani, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 11, November 2013*

2. **Software Defined Wireless Networks: Unbridling SDNs -** *Salvatore Costanzo, Laura Galluccio, Giacomo Morabito, Sergio Palazzo DIEEI –European Workshop on Software Defined Networking, 2012*

3. **Software-Defined Networks for Future Networks and Services -** *White Paper based on the IEEE Workshop, 29th January 2014*

4. **Software Defined Network Routing in Wireless Sensor Network-** *Junfeng Wang, Ping Zhai, Yin Zhang , Lei Shi, Gaoxiang Wu, Xiaobo Shi, Ping Zhou, 2016-17*

5. **Wireless Software Defined Networking: A Survey and Taxonomy-** *Israat Tanzeena Haque and Nael Abu-Ghazaleh, IEEE Communication Surveys, 2016-17*

6. **A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements-** *Hlabishi I. Kobo, Adnan M. Abu-Mahfuz, Gerhard P. Hancke, IEEE, March 2017*

7. **SD-WISE: A Software-Defined WIreless SEnsor network-** *Angelos-Christos G. Anadiotis, Laura Galluccio, Sebastiano Milardo, Giacomo Morabito, and Sergio Palazzo, IEEE, Oct 2017*

8. **On Scalability of Software-Defined Networks -** NetSoheil Hassas Yeganeh, Amin Tootoonchian, and Yashar Ganjali, University of Toronto IEEE Communications Magazine • February 2013

9. **Information-Centric Networking for the Internet of Things: Challenges and Opportunities** Marica Amadeo, Claudia Campolo, José Quevedo, Daniel Corujo, Antonella Molinaro, Antonio Iera, Rui L. Aguiar, and Athanasios V. Vasilakos IEEE Network • March/April 2016

10. **Cache "Less for More" in Information-Centric Networks** Wei Koong Chai, Diliang He, Ioannis Psaras, and George PavlouDepartment of Electronic and Electrical Engineering, University College London, WC1E 6BT, Gower Street, London, UK {w.chai,diliang.he.10,i.psaras,g.pavlou}@ee.ucl.ac.uk

11. **Energy-efficient algorithm based on multi-dimensional energy space for software-defined wireless sensor networks** Liao Wenxing; Wu Muqing; Wu Yuewei 2016 International Symposium on Wireless Communication Systems (ISWCS)

12. **A novel software defined wireless sensor network based grid to vehicle load management system** Nazmus S. Nafi; Khandakar Ahmed; Manoj Datta; Mark A. Gregory 2016 10th International Conference on Signal Processing and Communication Systems (ICSPCS)

13. **Software defined wireless sensor networks security challenges** Tebogo Kgogo; Bassey Isong; Adnan M. Abu-Mahfouz 2017 IEEE AFRICON Year: 2017

14. **Efficient Wireless Power Transfer in Software-Defined Wireless Sensor Networks** Waleed Ejaz; Muhammad Naeem; Mehak Basharat; Alagan Anpalagan; Sithamparanathan Kandeepan IEEE Sensors Journal Year: 2016

15. **Wireless multimedia sensor networks: A survey** Ian F. Akyildiz; Tommaso Melodia; Kaushik R. Chowdury IEEE Wireless Communications Year: 2007